Fast SSP Solvers Using Short-Sighted Labeling

Luis Pineda, Kyle H. Wray and Shlomo Zilberstein

College of Information and Computer Sciences, University of Massachusetts, Amherst, USA

July 9th

Motivation

- SSPs are a highly-expressive model for sequential decision making
- They can be used for decision-making in the presence of multiple goals

Caveat: solving SSPs optimally is a very computationally intensive task

Motivation

- SSPs are a highly-expressive model for sequential decision making
- They can be used for decision-making in the presence of multiple goals

Caveat: solving SSPs optimally is a very computationally intensive task

Motivation

- A range of model reduction and heuristic search techniques for solving SSPs are available
- But even restricting only to states in optimal policies can be prohibitive

Motivation

- Recent approaches attempt to reduce the reachable state space even more and use re-planning during execution
- However, they still have several drawbacks:
  - Restricted to particular problem representations (e.g., RFF and FF-Replan)
  - Involve pre-processing (e.g., $\mathcal{M}_l^k$ reduction)
  - Result in moderate reductions in time (e.g., SSiPP)

In this work...

- We introduce a new algorithm called FLARES (Fast Labeling from Residuals Using Samples)
- Modifies LRTDP to find high-performing policies much faster
- It can be extended to also find optimal policies

Stochastic Shortest Path Problems

An SSP is a tuple $\langle S, A, T, C, s_0, s_g \rangle$, where:

- S is a finite set of states
- A is a finite set of actions
- $T(s'|s, a) \in [0, 1]$ is a transition function
- $C(s, a) \in (0, \infty)$ is a cost function
- s_0 is an initial state
- s_g is a goal state

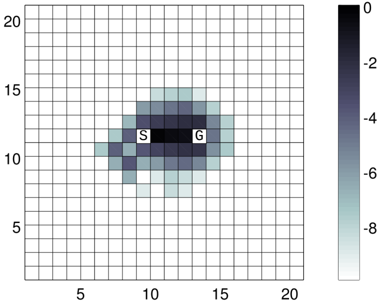
Solutions to SSPs

The optimal value function for an SSP can be found using:

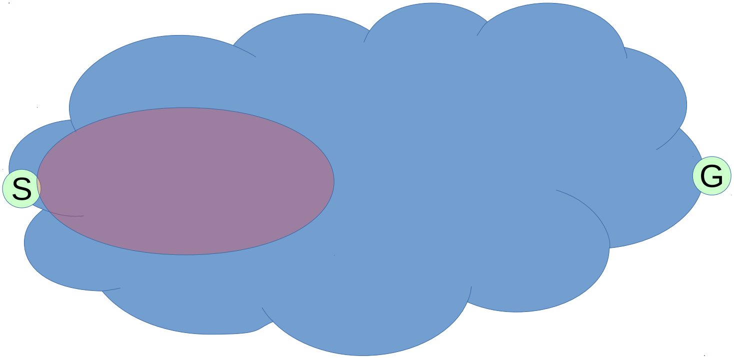
$$\mathrm{BU}(s) := \min_{a \in A} \left\{ C(s,a) + \sum_{s' \in S} T(s'|s,a) V(s') \right\} \tag{1}$$

$$\pi(s) = \arg\min_{a \in A} \left\{ C(s,a) + \sum_{s' \in S} T(s'|s,a) V(s') \right\} \tag{2}$$

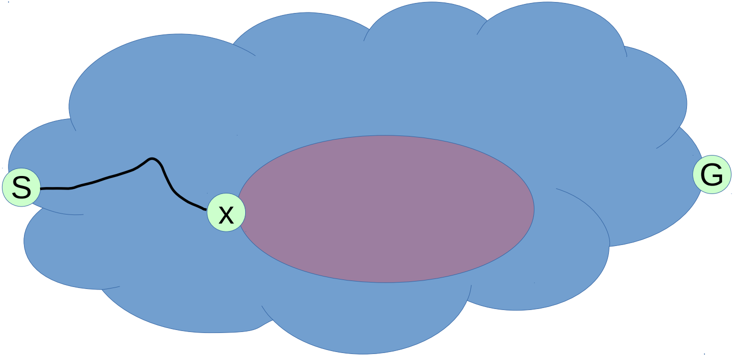
Problems with optimal heuristic search algorithms



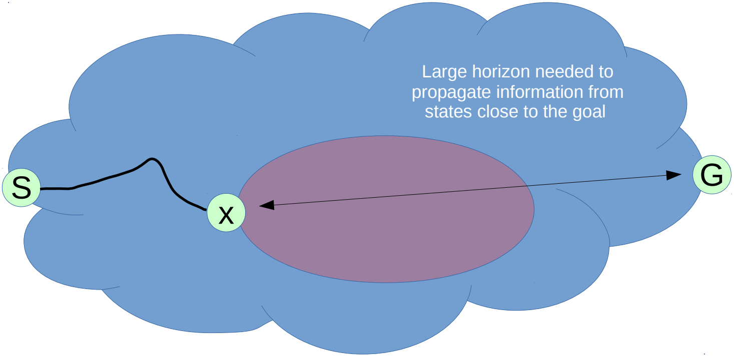
Depth-limited search



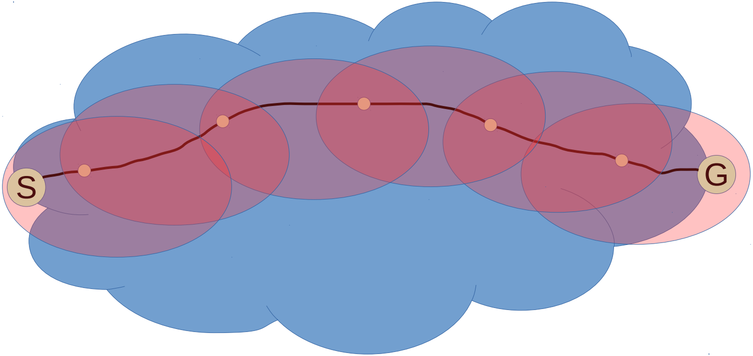
Depth-limited search



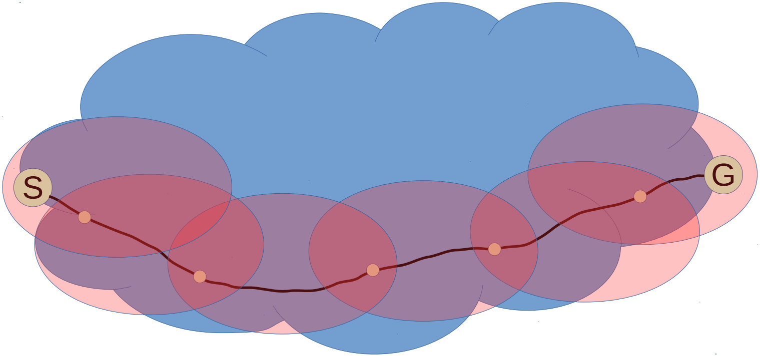
Depth-limited search



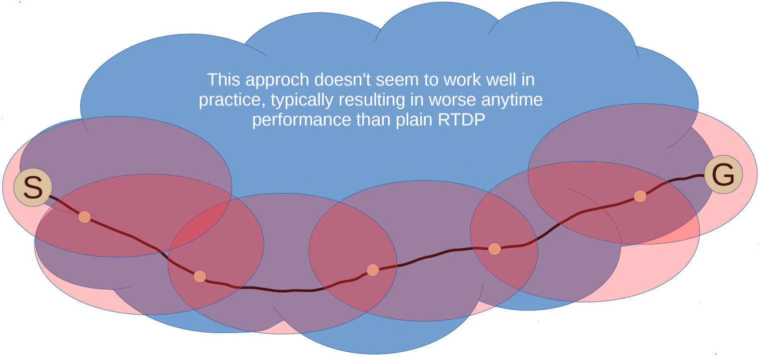
RTDP with short-sighted SSPs



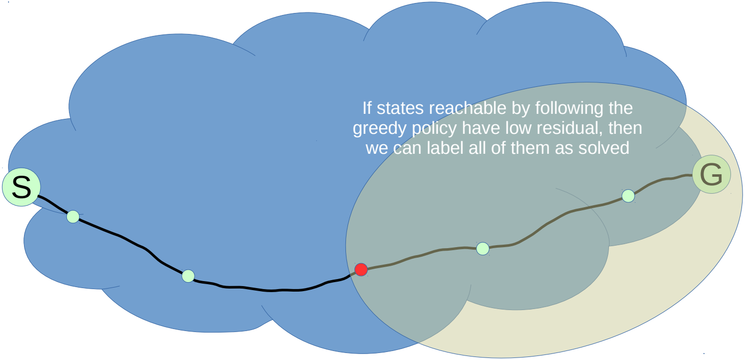
RTDP with short-sighted SSPs



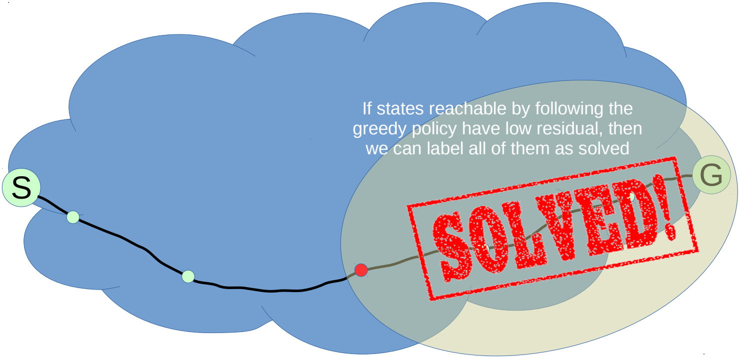
RTDP with short-sighted SSPs



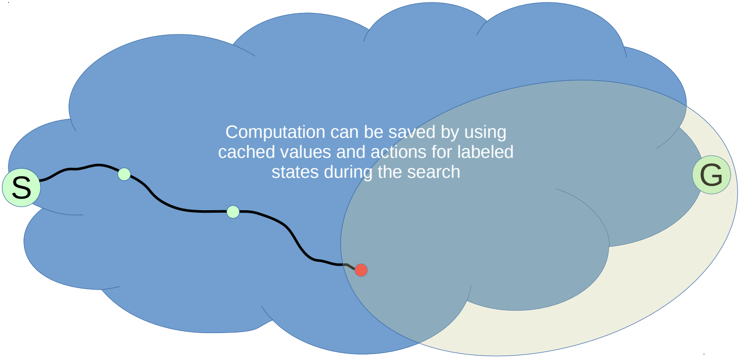
Labeling states (LRTDP)



Labeling states



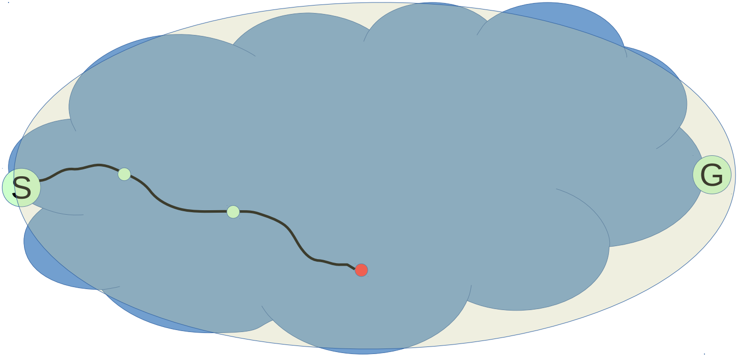
Labeling states



But...

...to label s, all states reachable from s must be checked...

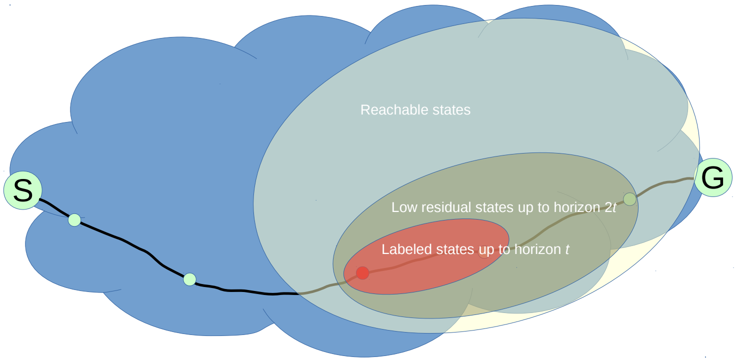
Labeling states can also be costly



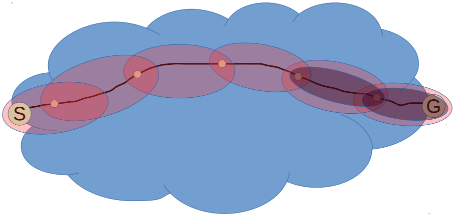
FLARES

Proposed approach: Move the short-sightedness to the labeling

FLARES

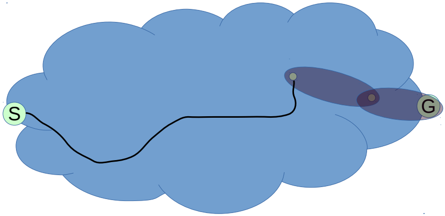

Reachable states
Low residual states up to horizon 2t
Labeled states up to horizon t
S
G

FLARES



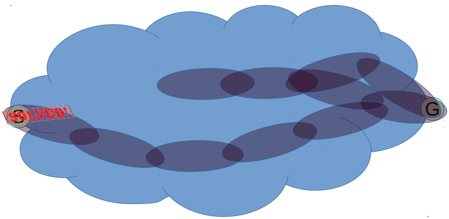
- Similar to the RTDP approach mentioned before...
- except that now computation can be reused and...
- there is a crisp termination condition that exploits short-sightedness

FLARES



- Similar to the RTDP approach mentioned before...
- except that now computation can be reused and...
- there is a crisp termination condition that exploits short-sightedness

FLARES



- Similar to the RTDP approach mentioned before...
- except that now computation can be reused and...
- there is a crisp termination condition that exploits short-sightedness

Correctness of labeling procedure

Proposition

FLARES labels a state s with $s.\text{SOLV} = true$ only if all states s' that can be reached from following the greedy policy satisfy $R(s') < \epsilon$.

Proposition

As long as a Bellman update of s' with $R(s') < \epsilon$ never results in $R(s') \geqslant \epsilon$, then FLARES labels a state s with $s.\text{D-SOLV} = true$ only if s is depth-t-solved.

Correctness of labeling procedure

Proposition

FLARES labels a state s with $s.\text{SOLV} = true$ only if all states s' that can be reached from following the greedy policy satisfy $R(s') < \epsilon$.

Proposition

As long as a Bellman update of s' with $R(s') < \epsilon$ never results in $R(s') \geqslant \epsilon$, then FLARES labels a state s with $s.\text{D-SOLV} = true$ only if s is depth-t-solved.

FLARES is guaranteed to terminate

Theorem

If the heuristic is admissible and monotone, FLARES terminates after at most $\epsilon^{-1} \sum_{s \in S} V^*(s) - V(s)$ *trials.*

An optimal version of FLARES

An optimal version of FLARES can be produced by calling FLARES multiple times with increasing horizon t

Theorem

If the initial heuristic is admissible and monotone, and ρ satisfies $\forall V, t,\ \rho(V, t) > t$, with $t_0 \geqslant 0$, then OPT-FLARES computes an optimal policy.

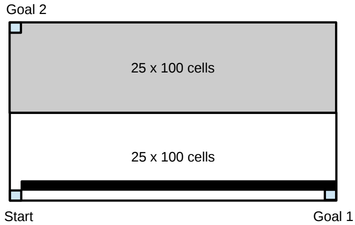
An optimal version of FLARES

An optimal version of FLARES can be produced by calling FLARES multiple times with increasing horizon t

Theorem

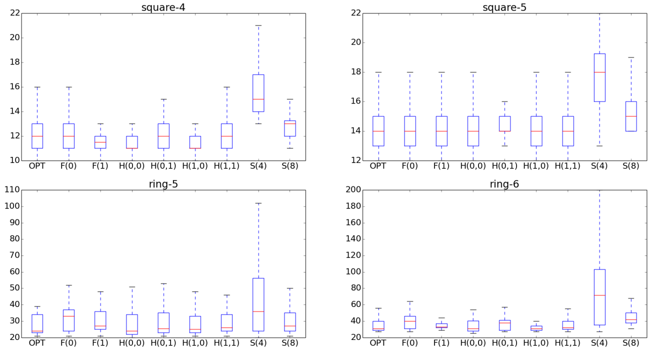
If the initial heuristic is admissible and monotone, and ρ satisfies $\forall V, t,\ \rho(V, t) > t$, with $t_0 \geqslant 0$, then OPT-FLARES computes an optimal policy.

Gridworld domain results



algorithm	cost	time
LRTDP	135	34.02
FLARES(0)	134.07 ± 0.84	0.586
FLARES(1)	135.63 ± 0.99	0.589
HDP(0,0)	208.9 ± 10.92	0.195
HDP(4,0)	135.22 ± 1.09	0.610
HDP(4,4)	133.91 ± 0.83	0.593
SSiPP(16)	441.12 ± 4.87	10.87
SSiPP(32)	400.87 ± 1.85	51.49
SSiPP(64)	136.49 ± 0.76	9.49

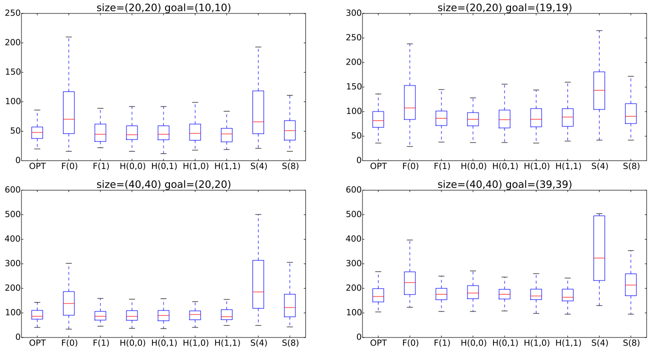
Racetrack domain - Average costs



Racetrack domain - Average planning time (seconds)

	square-4	square-5	ring-5	ring-6
LRTDP	49.89	262.84	11.64	65.02
FLARES(0)	**0.276**	**1.637**	**0.052**	**0.341**
FLARES(1)	**0.260**	**1.645**	**0.058**	**0.362**
HDP(0,0)	23.71	151.15	5.50	37.15
HDP(0,1)	26.84	145.12	5.84	37.08
HDP(1,0)	27.50	145.83	6.09	36.09
HDP(1,1)	28.41	142.02	6.10	38.27
SSiPP(4)	16.24	76.11	4.78	25.11
SSiPP(8)	48.61	178.98	20.13	89.72

Sailing domain - Average costs



Sailing domain - Average planning time (seconds)

	s=20 g=corner	s=40 g=corner	s=20 g=middle	s=40 g=middle
LRTDP	1.81	14.65	1.37	12.09
FLARES(0)	**0.33**	**3.15**	**0.138**	**1.142**
FLARES(1)	1.01	7.79	0.417	3.065
FLARES(2)	1.47	9.51	0.731	4.094
HDP(0,0)	1.33	11.93	0.854	7.034
HDP(0,1)	1.33	12.04	0.854	7.245
HDP(1,0)	1.35	11.85	0.853	7.133
HDP(1,1)	1.33	11.88	0.853	7.159
SSiPP(4)	3.05	8.83	1.60	5.69
SSiPP(8)	7.14	52.47	3.93	19.77

Conclusions

- We present an approach for short-sightedness in SSPs that applies it only for labeling
- Allows larger sections of the state to be explored and accelerate running times
- Based on this idea, we introduce a novel extension of LRTDP called FLARES
- Experimental results suggest that FLARES can produce near-optimal policies orders of magnitude faster than other state-of-the-art MDP solvers

Conclusions

- We present an approach for short-sightedness in SSPs that applies it only for labeling
- Allows larger sections of the state to be explored and accelerate running times
- Based on this idea, we introduce a novel extension of LRTDP called FLARES
- Experimental results suggest that FLARES can produce near-optimal policies orders of magnitude faster than other state-of-the-art MDP solvers

Conclusions

- We present an approach for short-sightedness in SSPs that applies it only for labeling
- Allows larger sections of the state to be explored and accelerate running times
- Based on this idea, we introduce a novel extension of LRTDP called FLARES
- Experimental results suggest that FLARES can produce near-optimal policies orders of magnitude faster than other state-of-the-art MDP solvers

Conclusions

- We present an approach for short-sightedness in SSPs that applies it only for labeling
- Allows larger sections of the state to be explored and accelerate running times
- Based on this idea, we introduce a novel extension of LRTDP called FLARES
- Experimental results suggest that FLARES can produce near-optimal policies orders of magnitude faster than other state-of-the-art MDP solvers

Conclusions

- We present an approach for short-sightedness in SSPs that applies it only for labeling
- Allows larger sections of the state to be explored and accelerate running times
- Based on this idea, we introduce a novel extension of LRTDP called FLARES
- Experimental results suggest that FLARES can produce near-optimal policies orders of magnitude faster than other state-of-the-art MDP solvers